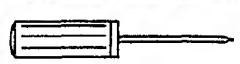
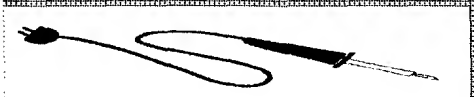
OS-9 Newsletter©

Volume IV Issue 7 | Bellingham OS-9 Users Forum | July 31, 1993

The "Correct All" Fix for the CoCo-3





COMMENT: The author assumes no liability for damage done to computing equipment. The subject of the article has been built and tested and has been working since 1987. I have made every attempt to double check my instructions and schematics for errors, but cannot gurantee they do not exist. This fix requires direct motherboard modifications and as such is highly susceptible to damage of said motherboard. Proceed with caution, and good luck!

* *

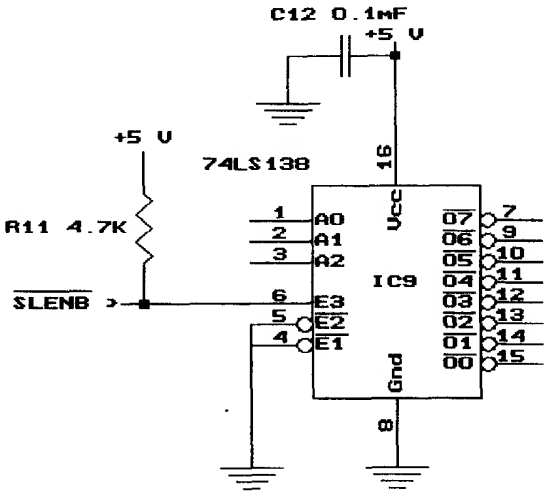
The CoCo III has a slight timing problem with SCS/CTS/ROM selects. This is due to IC9 (74LS138) being selected (i.e. transiting) during an entire E-clock cycle instead of being gated to the READ portion. The tech manual quotes "Due to the nature of the ROMs and in order to prevent data bus contention, the ROMs are enabled only during the E portion of a read cycle." pp 36. This is blatantly false when you look at IC9. It has both *G2A and *G2B and G1 mux selects tied wide open! The fix came from the Motorola 8-bit Microprocessor Handbook Data Sheets on the MC6883 SAM (GIME's father :-)).

Take a 74LS02 NOR gate and tie one input to pin 3 of IC9. **(See Figure 2)** Cut the connection between *G2A of IC9 (pin 4) and ground. Tie the NOR gate output to *G2A of IC9 (pin 4). Tie the other NOR gate input to the E-clock output at the intersection of R9 (47ohm resistor) and C10 (39pF capacitor). R9 is coming from pin 6 of the GIME chip. Make sure you tie pin 14 of the 74LS02 to pin 16 of IC9 (+5v) and pin 7 of 74LS02 to pin 8 of IC9 (Gnd).

This is not something for the weak of heart to try! At the very least you will have to remove the motherboard from the case (unless you clip IC9's little leg!) and cut a trace on the bottom of the mother-board. I cut/desoldered IC9, replaced it with a socket and built a little daughterboard which held a new IC9 plus 74LS02.

RESULTS:
This fix resulted in the following improvements on my system:

(1) Sparklies disappeared on graphics screens under OS9. (GIME chip circa 1986)
(2) Before the fix, I had blob problems. I no longer have the BLOB.



Original Schematic : INCORRECT!

Figure 1

*** IN THIS ISSUE ***

(3) My Performance Peripheal No Halt controller wouldn't run reliably in No Halt mode with the Burke & Burke RTC Hard Disk interface. They now coexist peacefully with the fix.

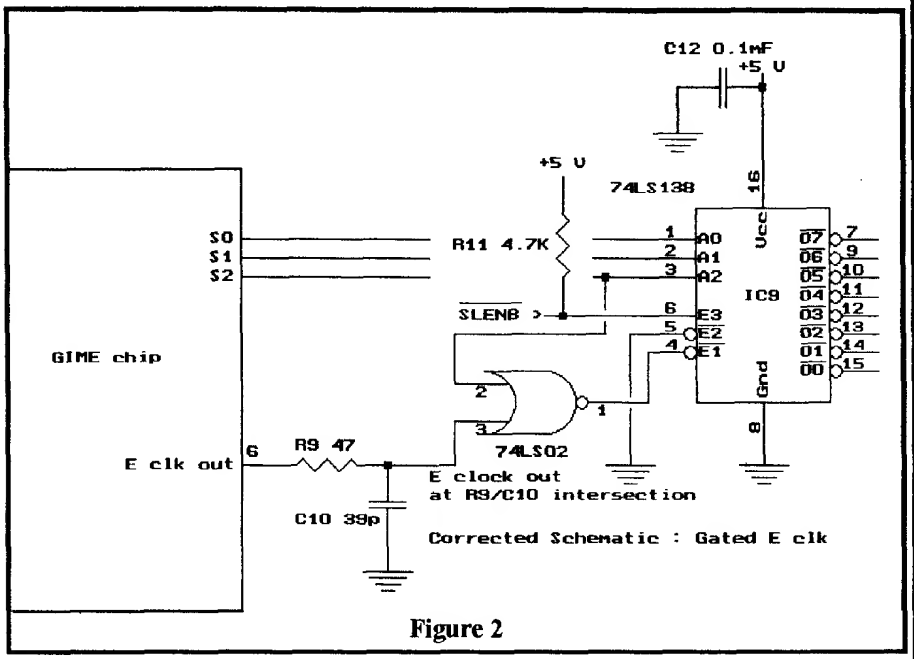


Figure 2

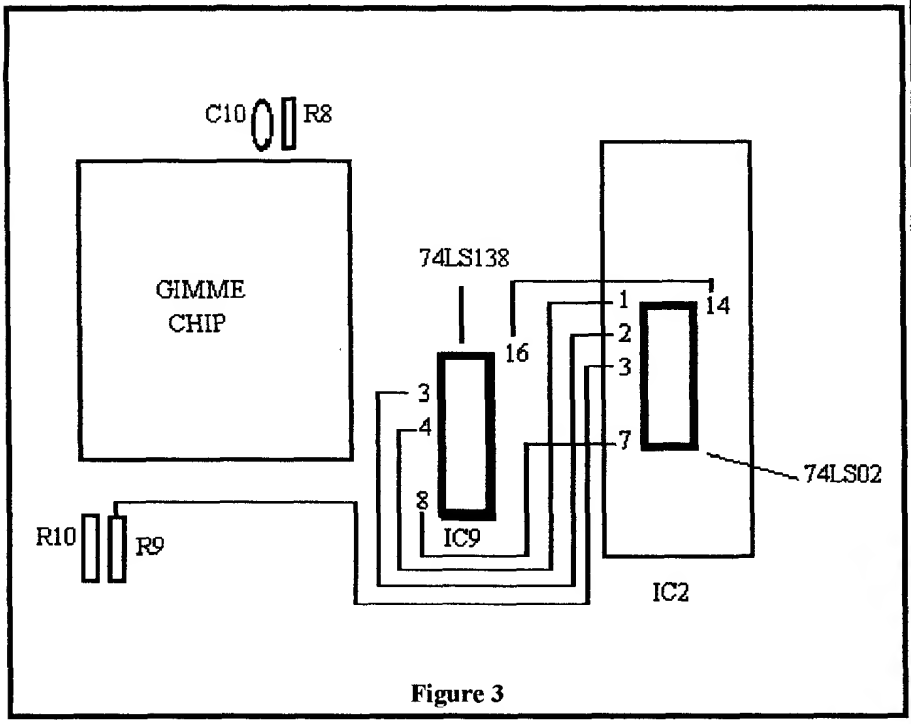


Figure 3

Note: *An article in "The Rainbow" said that Disto products "depended"? upon the timing not corresponding to the E clock. My personal opinion is:* ***bull twinkies!*** *As far as I know all Disto products should work with CoCo's 1 and 2 which do have the proper E-clock gating. Remember, that is just my opinion and all Disto users should be prepared for this fix not working. (If you put a socket in and build the daughterboard you can have it either way...)*

Charles C. Bundy IV
Internet; <COCO@PUCC>

OS-9 Newsletter

Editor: Rodger Alexander

OS-9 Newsletter is published monthly by the Bellingham OS-9 Users Forum and is protected under United States Copyright Laws. No material may be reproduced or copied in whole or in part without the expressed written permission of the Bellingham OS-9 Users Forum, 3404 Illinois Lane, Bellingham WA 98226

Submissions are welcomed in any format and can be mailed to the above address or sent via electronic mail to the editor: Rodger Alexander, on Delphi (UserID: SALZARD) or FidoNET (1:301/3401@fidonet.org) or Internet (ralexander@nwrdc.wednet.edu). Unfortunately, we do not have funds to reimburse authors of selected articles. However, a complimentary copy of the *OS-9 Newsletter* containing your article will be mailed to you, PLUS the satisfaction that you will have the admiration and appreciation of all of our readers.

The ***Bellingham OS-9 Users Forum*** is a hobbyist club, organized for the purpose of providing information, services, products and events that support the OS-9 operating system for 6809/68xxx based computers. Our efforts are not intended to earn or generate any profit for the club or any of it's members.

TO SUBSCRIBE

For 12 monthly issues of the *OS-9 Newsletter*, please send a US check or money order for $12 or $7 for a 6 month subscription. Mail your subscription order to:

OS-9 Newsletter
3404 Illinois Lane
Bellingham, WA 98226

Include your name, address and telephone number. You will receive your *OS-9 Newsletter* no later than the 10th of each month. Canadian orders, $13.50 for 1 yr. or $7.60 for 6 mo. Foreign orders $18 for 1 yr. or $10 for 6 mo.

Those Problematic Pointers

C Tutorial by Randy Kirschenmann

My last article hinted that most of the groundwork had been laid, and that we would soon be getting into some real coding examples. But, there remains one final topic that must be covered before we can jump in and do anything interesting... pointers.

Talk to any programmer new to C and ask them where they have the most problems and you'll get the same answer... pointers. Why are pointers so difficult? I suppose different people will give different responses to this, but basically, pointers can be confusing. With my background in assembler languages I had relatively minor difficulties in grasping this part of C, but non-the-less the pointer concept was probably the hardest part of the C learning curve, even in my case. For those programmers using languages without registers and relative addressing techniques, pointers can be very daunting, indeed.

So lets start at the beginning. We can declare a character data variable in C with the construct:

```
char c;
```

This, as we have seen will reserve a one byte area in memory, which can be accessed using the given label, *c*. To declare a pointer to that data type we would use:

```
char *cptr;
```

The name (cptr) is arbitrary, but the asterisk (*) signals the compiler that the variable we created is not a one byte data item, but a data type large enough to hold the address of a char type. On the CoCo III, under OS-9 Level II, with Microware's C compiler, this will be a two byte data item. As declared above, this variable has no value assigned to it, and cannot be used yet. To place the address of c in this pointer we use the notation:

```
cptr = &c;
```

Now the variable *cptr* holds a valid address, that of the data item, *c*, and we can use it to access the variable *c* until its value (cptr's value) is changed. In order to access *c*, via this pointer, rather than the pointer variable, we use the notation for dereferencing a pointer:

```
*cptr = 'a';
```

In this statement I have assigned a value of '*a*' to the variable *c*. The pointer variable, *cptr*, still points to *c*; its value hasn't changed. If, at this point in our program, we were to call the *printf()* function:

```
printf("%c", c);
```

our display terminal would show the character "*a*". Go ahead and test this. The complete listing would be:

```
main() {
   char c;
   char *cptr;
   cptr = &c;
   *cptr = 'a';
   printf("%c", c);
}
```

For a complete description of the *printf* function you should look in the C Compiler Manual. What we have done here is ask *printf* to output a character variable, denoted by %c, called *c*. I could have used a better choice of variable names to avoid all of this confusion.

In our example we have shown how to declare a pointer, how to assign a value to it (an address) and how to use the pointer to access the storage address to which it was assigned. This is very basic stuff and probably doesn't seem all that complicated. However, this is just the tip of the iceberg. Before we get into some of the more complicated aspects of pointers, I need to point out that a pointer to *char* can only be used to point to char data types. Pointers to *int, long int, float,* etc. need to be declared individually if we need to point to any other data types. This is important to remember. Pointer arithmetic is possible (adding integers to pointers, subtracting one pointer from another) because the compiler "knows" how many bytes each of the data items takes up in memory.

Well, now that I've mentioned pointer arithemtic, I guess I'ld better go ahead and explain a bit further what I'm talking about. First of all, some preliminaries. Let's define an array of data type *int*:

```
int i[10];
```

The array name, *i*, can be considered as a pointer. The only difference in this is that *i* is a pointer constant, not a pointer variable; that is, we cannot assign to *i* any other value. It contains the address of the first element of our array... *i[0]*. We can access the second element of this array by coding *i[1]*. Also, we can write *i+1* to provide access to this element. Similarly, *i+9* accesses the last element of the array. This is pointer arithmetic in its most elementary form. What do we access with the construct *i+10*, or *i-1*? These values are undefined. That means that what the compiler does if you code something like this is compiler dependant, and often unpredictable.

Now let's define a couple of pointers to int and an int:

```
int *iptr1, *iptr2, j;
```

and then point them to something useful:

```
j = 5;
iptr1 = +1;
iptr2 = i+j;
```

we can then use the construct *iptr2 - iptr1* to count the number of elements that lie between the two pointers. Also the expressions *iptr1 > iptr2* and *iptr1 < iptr2* are meaningful and will be properly evaluated. This is the extent of pointer arithmetic, but many useful things can be accomplished as long as the rules are followed. Pointers which do not point to arrays, or point to different arrays cannot be compared or subtracted. Integer values, when added to or subtracted from pointers, which would cause the pointer to fall outside the bounds of the array cannot be used. Using this technique we can implement a simple array iterator routine:

```
char arr[] = "Hi! I am an array of characters";
main() {
    char *cptr = arr;
    while(*cptr)
        printf("%c\n", *cptr++);
}
```

In this example, I have declared a global array of characters and initiallized it with the string, *"Hi! I am an array of characters"*. The array needs to be defined globally (declared outside of the body of the function) in order for the compiler to initiallize it for us. If we were to declare it within the body of the function (main) we would have to explicitly initiallize the array by using *strcpy()* or some other means to place the character values we want into the array. Global variables and constants can be accessed by all functions within the program, whereas variables and constants defined within a function can only be accessed by that function.

Next, I have defined a char pointer, *cptr* and given it the address of our array. Remember, our pointers cannot be used until they hold a valid address. The first thing to look for when your programs do weird things is for the use of pointers that haven't been properly initiallized. The real work of our example is found in:

```
while(*cptr)
    printf("%c\n", *cptr++);
```

The while loop is checking for true or false. By coding *while(*cptr)* we are asking if **cptr* is true. In C an expression is true if it evaluates to a non-zero value, and false if it evaluates to zero. Since our pointer is defined to point to the char array, *arr*, and since we are evaluating **cptr*, which is the character to which it points, when evaluated the first time our loop looks at 'H'. This is hexidecimal 0x48; non-zero. Therefore our program will execute the body of the loop, which is one statement:

```
printf("%c\n", *cptr++);
```

to output the character pointed to by cptr. The expression **cptr++* advances the pointer by the length of a char data type, one byte, to point to the next character in our array. The ++ following *cptr means that this advance will take place following the use of the pointer in the printf function. If we had coded:

```
printf("%c\n", ++*cptr);
```

then the pointer would be advanced before used in printf(), which would cause the char *'i'* to have been displayed instead of *'H'*. Following the execution of the body of the loop, control returns to the top of the loop to evaluate the while expression again. This time, since our pointer had been incremented, the value at *arr[1]* is examined for zero.

Our loop continues to execute until a zero value is found which will be placed in our array by the C startup code generated by cstart.r when our source file is compiled and linked. This will cause the loop to end, skipping the body of the loop, and reaching the end of our main function, which will return control back to the caller... the OS-9 operating system. I recomend entering these short examples, compiling and running them to get a feel for C and the use of the compiler.

Instead of using the while loop we could've used the for loop. I will leave as an exersize the substitution into the body of our program. As a hint I'll mention that the initiallization of our loop iterator may be empty.

Our discussion of pointers is not very extensive. There remains many aspects of pointers that cannot be covered in this short discussion. We will be seeing much more on pointers in future articles.

==Randy==

PNW CoCoFEST Photo Album

Chet Simmons, RS-Dos Graphics

Interactive Media (Part II)

Controling the Pioneer Laser Disk Player with Basic09

Last month I left the last paragraph hanging..... "In the mean time, check out your RS-232 Pak to make sure it works on a CoCo-3. If not, replace the 6551 PIA ." The reason it may be necessary to replace the PIA chip in your Deluxe RS-232 Pak is that the original 6551 PIA chip was not designed to work at 2MHz. There is a 2MHz replacement with the nomenclature of **6551A.** I had trouble finding this replacement chip, but once again, **Terry Laraway** came to the rescue with a supply at $4 each.

So now that I have a working 2MHz RS-232 Pak and a complete list of all of the ***Control Commands*** for the Pioneer 2000 series Laser Disk Players (See last month's installment), all that is needed is an easy to use menuing program that prompts the user for the desired action. Figure 3 is what the Menu looks like on an OS-9 System (I also wrote a similar program for MS-DOS). Notice that I have only included the very basic options, but after getting familiar with the Basic09 listing you will see how easy it is to add whatever options you want to include.

Figure 4

This is still an unfinished project. The menuing environment makes operation of the Laser Player very user friendly, but in the long run does not offer any real advantage over the hand held remote control unit. The ultimate goal is an interactive computer program, or a Basic09 listing that will program the laser player to store selected command sequences and execute those commands in the proper sequence, either automatically or by just pressing the enter key or pressing the PLAY button on the remote control. Anyone with any programming experience in any language can see that this would not be a very difficult task.

The program would be divided into two main parts or procedures: One for programming in the sequences using INPUT statements for storing the desired actions into variable strings, and the other for the play-back of the programmed sequences by executing the variables in the desired order. Simple!

==Rodger Alexander==

```
PROCEDURE menu
   REM Menuing Control for Laser Disk Player
   DIM out_path:BYTE
   DIM command,numbers,laser:STRING[8]
   DIM menu:STRING[1]
   DIM frame,video,audio:BOOLEAN
   frame=FALSE \video=TRUE \audio=TRUE
   laser="/T2"
   OPEN #out_path,laser:WRITE
10 PRINT CHR$(12)
   PRINT USING "s79^","LASER PLAYER SELECTION MENU"
   PRINT  \ PRINT
   PRINT TAB(30); "<P>lay"
   PRINT TAB(30); "<S>top"
   PRINT TAB(30); "<F>ind Frame #"
   PRINT TAB(30); "<S>till Frame"
   IF frame=FALSE THEN PRINT TAB(30); "<D>isplay
Frame ON"
   ELSE PRINT TAB(30); "<D>isplay Frame # OFF"
   ENDIF
   IF video=FALSE THEN PRINT TAB(30); "<V>ideo
Display ON"
   ELSE PRINT TAB(30); "<V>ideo Display OFF"
   ENDIF
   IF audio=FALSE THEN PRINT TAB(30); "<A>udio
(Sound) ON"
   ELSE PRINT TAB(30); "<A>udio (Sound) OFF"
   ENDIF
   PRINT TAB(30); "<O>pen Drawer"
   PRINT TAB(30); "<C>lose Drawer"
   PRINT TAB(30); "<Q>uit Program"
   PRINT
   PRINT TAB(30); "Enter Your Choice: ";
   GET #0,menu
   IF menu="P" OR menu="p" THEN
     PRINT #out_path,"PL"
   ENDIF
   IF menu="S" OR menu="s" THEN
      PRINT #out_path,"ST"
   ENDIF
   IF menu="F" OR menu="f" THEN
     PRINT  \ PRINT
     PRINT TAB(30); "Type in Frame #: ";
     INPUT numbers
     command=numbers+"SE"
     PRINT #out_path,command
   ENDIF
   IF menu="S" OR menu="s" THEN
     PRINT #out_path,"ST"
   ENDIF
   IF menu="D" OR menu="d" AND frame=FALSE THEN
     PRINT #out_path,"1DS"
     frame=TRUE
   ELSE
     IF menu="D" OR menu="d" AND frame=TRUE THEN
       PRINT #out_path,"0DS"
       frame=FALSE
     ENDIF
```

(Continued on page 8, col. 2)

Q: How can I connect a null modem between my Mac Plus and my Color Computer 3?

==Dick Little==

A: Here are suggested wiring configurations for a null modem cable between Mac Plus style 8 pin mini DIN connector and Radio Shack Color Computer (bit banger and RS232 pak).

This is actually a "first guess" on the correct wiring for a null modem cable between a Radio Shack Color Computer and Mac Plus style 8 pin mini DIN RS232 connector. I'm basing it on my info for pin outs for Mac to Modem cables and on my pin out info on the Mac RS422 connector. This may or may not work! Please let me know if you try this about what success you have, and of any corrections you discover.

Mac 8 pin Mini DIN	CoCo 3 4 pin DIN
1,2	1
3	2
4,8	3
5	4

ALSO: Connect pin 1 to pin 2 on the Mac mini DIN and Connect pin 8 to pin 4 on the Mac mini DIN

Mac 8 pin Mini DIN	CoCo RS232 pak DB 25
3	3
4	7
5	2
1,2	4,5
4,8	6,8,20

ALSO: Connect pin 1 to pin 2 on the Mac mini DIN. Connect pin 8 to pin 4 on the Mac mini DIN. Connect pin 4 to pin 5 of the RS232 pak DB 25 connector and Connect pins 6,8, and 20 together on the RS232 pak DB 25.

Hint:
If these cables don't work, try reversing the wires going to pins 4 and 2 of the CoCo 4 pin DIN connector, or try reversing the wires going to pins 2 and 3 of the DB 25 connector, and see if that fixes things.

Background Data:
Pin Out for Mac RS422 port:
1 Handshake Output (DTR from Mac)
2 Handshake Input (to Mac)
3 TxD- (negative data out of balanced data lines) (from Mac)
4 ground
5 RxD- (negative data in) (to Mac)
6 TxD+ (positive data out)
7 ??
8 RxD+ (negative data in)

Notes:
When using the RS422 port with an RS232 port:
(1) Use TxD- and RxD- as you would RxD and TxD on a CoCo or IMB type computer.
(2) Hook RxD+ to ground on the Mac

==Marty Goodman;Delphi==
OS-9 Forum Sysop

Delmar adds another 68030 to the OSK Market

The SYSTEM V OS9 computer system provides a flexibility unavailable heretofore for systems running OS9. Building on the design concepts proven in the SYSTEM IV computer system, the SYSTEM V adds a new dimension using modular construction. The microprocessor, math co-processor and necessary 'glue' chips are on a small, sepa rate microprocessor (daughter) board which plugs into the main or system board. The first microprocessor board designed is for a 68020 microprocessor running at 25 MHz and optionally, 33 MHz.

FEATURES
- Upgradeable board design
- Up to 128 MB Simm memory
- 2K Battery-Backed up SRAM (32K optional)
- 0 wait-state memory up to 25 MHz.
- On-board IDE Controller
- On-board SCSI Interface
- 32-bit data and address buses
- 5 16-bit and 2 8-bit slots on separate ISA bus
- Sound Port using 8-bit DAC
- Floppy controller support for 360K to 1.44 MB drives
- 4 on-board serial ports
- Printer port and second parallel port
- SVGA (1024 x 746 x 256) video card (optional)
- GWINDOWS with DESKTOP (optional)
- XT Size board (8.5" x 12")
- MC68882 Math Coprocessor (optional)
- 16K - 128K EPROM
- Professional OS-9, version 2.4 with K&R C-Compiler, Editor, Assembler, Linker, over 80 Utilites and MW manuals
- Ultra-C Compiler (optional)
- SYSTEM V Users Manual with schematics
- Mini-Tower case with 200 watt power supply
- Fully tested - ready to run

For more information, contact: Delmar Co
PO Box 78 - Middletown Plaza - Middletown, DE 19709
302-378-2555 - FAX - 302-378-2556

Review Public Domain Files

I have not done any Public Domain File Reviews for quite a while simply due to the lack of time needed to download, unarchive and play around with all of the various programs that have been posted on Delphi and other Bulletin Boards. However, I have come across a few real gems that I would like to share with you that you might seriously consider adding to your collection.

MOTOR:
This is a small binary utility that operates the cassette relay on the CoCo. The archived files also includes a patch file to fix an interfering bit in the PRINTER module and a sample diagram in vef format of how to use the cassette port to control large voltage devices, such as turning off your hard drives while your bulletin board is idle, etc. The utility is posted under the name of *PRINTER.AR*, which is a slight misleading.

SOUND:
This is a cutie! It's a replacement for "display 7", but with a few enhancements. The command accepts a combination of two arguments: **u** for a tone that quickly slides up, and **d** for a tone that quickly slides down. You can repeat the arguments as many times as you want to lengthen the duration of the tone.

Example: **sound uuuuuu** OR **sound udududududud**
The result is an arcade sounding warble that is at least more interesting then the standard **"Beep"**

WALK:
This is another cutie! It's an animated graphic icon of a little stick figure that walks around the screen with an occasional display of the time and date or some other short message, such as "GET AWAY FROM THE SCREEN", etc. It is designed to be used as a screen saver, but alas, it is not self activating like a true screen saver should be.

COLORC:
This is a file transfer utility for OS-9 and PC Formats. However, this one is to be used on an IBM, not the CoCo. This could be the answer for those having trouble with the *PCDOS* utility.

MBACKUP:
This is a backup utility that overcomes the 64K ram block problem associated with the Color Computer. In fact, if you have 1 or 2 Megs of RAM you could do a backup of a 720K disk in one pass. The command syntax is as follows:

OS9:> **mbackup -b=720 /d2**

The one terrible thing about this utility however, is that it is a single drive backup only.

LZH10:
OS-9 users are now faced with a multitude of choices for archiving files. Before, only *AR* and *PAK* were available as archiving utilities. But now, the entire gambit of MS-DOS archives have been ported over. You now have the choice of ***ZIP, ARC, ARJ*** and ***LZH.*** Unfortunately, there are several variations of the *LZH* archive/de-archive utility, and there are several examples that have been ported over to OS-9. ***LZH10*** is the only version that I found able to unarchive every <filename.LZH> that I could find. I did find some "quicker" versions but they were not able to de-archive all of the archived samples. This version of *LZH* was ported over by Matthew Thompson.

UNARJ09:
I have included this utility in my list, only because it was the only *ARJ* archive/de-archive utility I have found for OS-9. Believe it or not, I have come across two files that were archived by this newest of MS-DOS archiving utilities in OS-9 databases. The *ARJ* format is suppose to be slightly more efficient then the *LZH* version, so we may see more and more of this type. Unfortunately this utility only de-archives, it will not archive to the *ARJ* format.

CUTS and UUENCODE:
If you do any telecommunications work with your OS-9 System, both of these encoding/decoding utilities are a necessity. What they do is take binary files that you would normally transfer using an error checking protocol such as XMODEM or YMODEM, etc., and convert them to an ASCII format. *(See the Q&A article in this issue re: BINEX.)*

Why would you want to convert a binary file to ASCII format? Well, what if you wanted to transfer a file over a communications network that doesn't provide any error checking protocol support for transferring binary files. Another example would be sending a binary file via Electronic Mail to an individual. It can't be done.

Ever notice what happens to your OS-9 screen when you try to list a binary file? The screen goes crazy and sometimes the computer even locks up. *CUTS* and *UUENCODE* encode binary files into individual lines of ASCII/ANSI code that will "list" harmlessly through the electronic mail networks.

Case in point! The schematic diagrams included in the ***Correct All Fix*** article in this issue were included in the text file. Think about that for a minute. How can you include a GIF graphics file within a text file? Answer: The GIF files were ***UUENCODE***d into ASCII format. When I received the file, I cut the encoded section out using a block save command in my text editor, and then I ***uudecode***'d the ASCII graphics file back to it's original GIF format.

==Rodger Alexander==

CoCoPRO Software

STILL AVAILABLE

Rick Cooper of *Rick's Computer Enterprise* (CoCo Friend's Disk Magazine) has purchased a lease of the ***COCOPRO***'s Software. Here is a listing of what he has and his special packaged sale prices:

Data Windows	*$39.95*
Multi-Menu	*$14.95*
	together: $15 + $2 S/H
Newspaper 09	*$34.95*
News Fonts	*$ 9.95*
The Zapper	*$14.95*
	together: $15 + $2 S/H
OS9 Level II BBS	*$19.95*
Disk Manager Tree	*$18.95*
Tools II	*$24.95*
	together: $15 + $2 S/H
Level II Tools	*$19.95*
Presto-Partner	*$19.95*
Data Merger	*$12.95*
	together: $15 + $2 S/H

SUPER DEAL---ENTIRE PACKAGE $55 + $5 S/H

Laser Control Menu Listing

(Continued from pag. 5)

```
ENDIF
   IF menu="V" OR menu="v" AND video=FALSE THEN
     PRINT #out_path,"1VD"
     video=TRUE
   ELSE
     IF menu="V" OR menu="v" AND video=TRUE THEN
       PRINT #out_path,"0VD"
       video=FALSE
     ENDIF
   ENDIF
   IF menu="A" OR menu="a" AND audio=TRUE THEN
     PRINT #out_path,"0AD"
     audio=FALSE
   ELSE
     IF menu="A" OR menu="a" AND audio=FALSE THEN
       PRINT #out_path,"1AD"
       audio=TRUE
     ENDIF
   ENDIF
   IF menu="O" OR menu="o" THEN
     PRINT #out_path,"OP"
   ENDIF
   IF menu="C" OR menu="c" THEN
     PRINT #out_path,"CO"
   ENDIF
   IF menu="Q" OR menu="q" THEN
     PRINT CHR$(12)
     CLOSE #out_path
       END
     ELSE
```

Great Stuff for your OS-9 System

We've been in the software business for over 10 years--and we've developed lots of excellent software over that time. We don't have room in this space to tell you everthing, but we'd love to send you our catalogue listing all of our products. Great stuff like our ***Ved*** text editor, ***Vprint*** text formatter, ***Cribbage***, ***Magazine Index System***, ***Ultra Label Maker***, ***Vmail***, amd more.

So you only get what you need, please specifiy OS-9 or OS9/68000!

Bob van der Poel Software

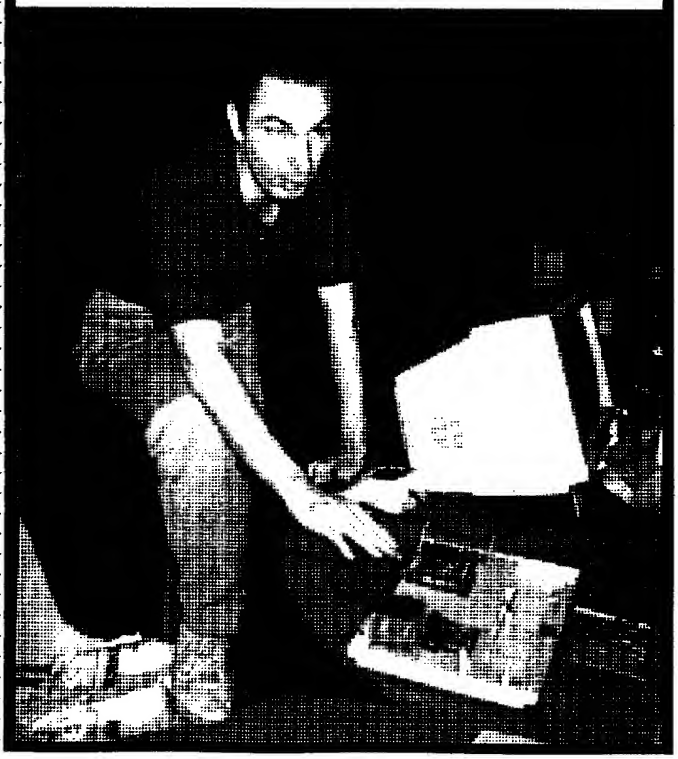
PO Box 355
Porthill, ID
US 83853

PO Box 57
Wynndel, BC
Canada VOB 2NO

Phone (604)-866-5772

PNW CoCoFEST Photo Album

Chris Burke, "The Rocket"



Pin Outs for 512K Board Connectors

Orientation and Pin Numbering:
"Rear" means toward the back of the Color Computer 3, where its joystick and other connectors are. "front" means toward the keyboard. "left" and "right" are used with the assumption you are facing the computer with the keyboard immediately in front of you, and the part of the computer with the four DIN and other connectors away from you, as you normally would face the computer were you about to use it.

The 512K memory board connector consists of three 12 pin connectors. CN4 is located just to the rear of the keyboard connector (CN2), just in front of IC12. What I call "pin 1" of CN4 is the pin nearest pin one of the keyboard connector, or the pin at the left hand edge of the connector. CN6 and CN5 are adjacent to each other. They look at first glance like one 24 pin connector, not the two separate 12 pin connectors that they actually are. They are both to the left of the four 18 pin 64K by 4 DRAM chip sockets, with CN6 more toward the rear than CN5. Pin 1 of CN6 and CN5 are understood to be the pins that are most rearward on those connectors. That is, pin 1 of CN6 is closest to the six pin joystick connector JK1, and pin 1 of CN5 is closest to CN6. All three connectors (CN4,5, and 6) are clearly labelled as such on the component side of the printed circuit board in white silk screen characters. However, pin 1 of these connectors is not clearly marked, hence my detailed description of what *I* am calling "pin 1" of each connector. I've tried to be consistent with the way other parts on the circuit board are numbered.

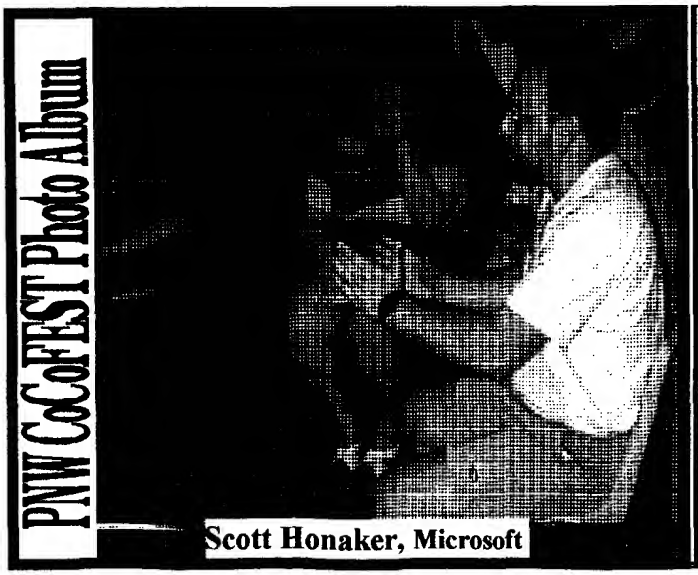
NOTES:

(1) Pin 11 of CN4 is for the ADDED address line required when one upgrades from 128K to 512K. Thus, all the OTHER address (and data and timing control) lines are used on both 128K and 512K CoCo's, but pin 11 of CN4 represents a line used ONLY on 512K CoCo 3's. For those hackers attempting a weird four-stack piggyback memory upgrade, note that pin 11 of CN4 is connected to a plate-thru solder pad just in FRONT of the hole for pin 11 on CN4, and so a wire can easily be attached to it there.

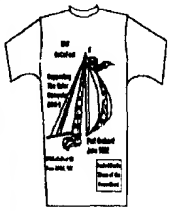
CN4			CN5		CN6	
pin #	function		pin #	function	pin #	function
1	gnd		1	gnd	1	gnd
2	*RAS		2	data	2	+5 VDC
3	adr		3	data	3	data
4	adr		4	data	4	data
5	adr		5	*WE1	5	data
6	adr		6	data	6	data
7	adr		7	*CAS	7	data
8	adr		8	data	8	data
9	adr		9	data	9	data
10	adr		10	data	10	data
11	adr	-A8-	11	data	11	*WE0
12	gnd		12	gnd	12	gnd

(2) All of the "data" lines are D/Q lines. That is, when used with a 512K upgrade and 41256 chips, the D and Q pins of each chip are shorted to each other to make up one "data" line. The original 4464 chips used in the 128K configuration have pins that already ARE D/Q lines to begin with.

(3) CoCo 3 memory is addressed as two 8 bit wide banks. These two banks are separated using the Write Enable line (*WE) on the chips. Chips contributing the first eight data lines have their *WE wired together as *WE0, and those contributing the other eight data lines have their *WE lines wired together to make *WE1. All of the data lines on CN5 are from chips whose *WE line become *WE1, and the data lines on CN6 are from chips whose *WE's become *WE0.



The PNW CoCoFEST III was a BIG success. We have a few T-shirts and Collector mugs left. We want to liquidate them for seed money for advertising and pre-FEST costs in '94.



While supplies last, order your T-shirt (specify M, L, XL, or XXL) & the 3rd annual collector's mug, "*The History of Computer Chips*". Send only $19 (US) to Port O' CoCo Club, 3046 Banner Rd SE, Port Orchard, WA 98366-8810.. Because of UPS your address must be to a physical location, not a P.O. Box. Checks will be returned when we run out.

PNW CoCoFEST IV will be June 24-25, 1994
Mark it on your calendar, NOW!

Club Activities Report

Bellingham OS9 Users Group - Longview/Kelso CoCo Club
Mt. Rainier CoCo Club - Port O'CoCo Club - Seattle 68xxx Mug

Bellingham OS9 Users Forum

The Bellingham OS-9 Users Forum (notice the name change) had some "responsibility" task to take care of at their July meeting. Since the "Forum" is responsible for the content in *Newsletter* and the *Tutorial (and upgrade manual)*, updating and corrections need to be done ASAP so as not to upset those who purchase the club's products.

The first item was to fix the parallel printer port posted in the February '92 issue of the *Newsletter*. This project appears to work fine with a CoCo 1 or 2 using OS-9 Level I, but causes interference with the Keyboard on the CoCo-3. Rodger has been seeking solutions to the problem for the past several months. **Gerry McCleary** from Calgary, BC Canada came up with the solution. The modifications to make this project work on the CoCo-3 will be posted in the August issue of the *Newsletter*.

Buzz Jones, from the Port O'CoCo Club, called regarding the articles in the May and August '92 issues on how to make a Hard Drive interface. In the August issue (and the *Tutorial)* an inadvertent trace line was drawn from the input of the first OR gate to the input of the second OR gate. The output of the first OR gate was blank. Obviously, the output of the first OR gate should connect to the input of the second OR gate since the input on the first OR gate is already connected to the CoCo's SCS line on pin 36 of the CoCo's I/O port. There was also a misprint in the labeling of the NMI line which was printed as Vmz. A corrected diagram will be posted in the next issue of the *Newsletter*.

We hope that the above bloopers have not caused anyone any great grief. I know how frustrating it can be to build something and then not have it work. We will try to do better in the future. BTW (By The Way), the article on page 1 of this issue of the *Newsletter* has been tested and is installed on two of our CoCo's with results as reported by the author. A very good project and extremely easy to do.....about 15 minutes work and about $1 in parts.

==Barbara Alexander==

Mt. Rainier CoCo Club

Our July meeting, held on the 8th at the Fernhill Library, at South 72nd and Yakima, was opened a few minutes later than usual by our new president, **John Schliep**. We had a whopping attendance of nine! New additions to our membership included John's father-in-law, **Jim Sparks**, and his wife who are CoCo II users. **Mrs. Sparks** came looking for a database for her recipe files, and was led to the VIP series of programs, which run on CoCo II's as well as CoCo III's, by **Alan Johnson**. But more on that later.

First off, the meeting started with our visitor from the Port 'O COCO, **Don Zimmerman.** Don gaves us his personal impressions of the recent and exciting PNW-COCOFEST, held in Port Orchard. Also, Don brought some of the remaining mugs and T-shirts and offered them for sale to any interested members. I believe there are yet a limited number of these available, and anyone interested should contact Don.

Following Don's presentation, **Randy Kirschenmann** gave a presentation on screen input methods in C. A short example program was presented which demonstrated how different function calls to obtain data from the keyboard worked. First, the *getchar()* function was shown. This function call will accept data from the keyboard and delay processing it until it receives the end-of-line character, *0x0D*, which is sent when the RETURN key is pressed. Next was shown a home-brewed function, *getch()*, which uses a call to the OS-9 system call *I$READ* to obtain data directly. This function was shown to process the characters immediately without waiting for <RETURN> to be pressed. Next, Randy showed how echoing to the display terminal could be shut off within the C program, to give complete control of both input and output to the screen to the programmer.

Finally, **Alan Johnson** demoed the VIP data base system. He showed us how to start the program and select the desired controls. This was shown on both a CoCo II and a CoCo III. What was interesting to me in this, was that both CoCo's were hooked up simutaneously to the same monitor, a Magnivox Professsional 80 815. This monitor had two sets of input cables: one that utilized the standard RCA jacks, the other which accomodates the CoCo III's display cable. There's a switch on the monitor to select from either of these input cables. After booting VIP on both machines, a simple throw of the switch would demo either system. This made for a very interesting comparison of the same software running on two distinct platforms.

As mentioned before, Mrs. Sparks was looking for a recipe database... Well, needless to say, she found a very good one that runs on her CoCo II. We hope to hear a report on her progress at our next meeting.

Our meetings are held each month, on the second Thursday, at 6:45 PM, at the Fernhill Library, on the corner of South 72nd and Yakima, in Tacoma. We hope to see any of you that can make it there.

==Randy Kirschenmann==

Port O CoCo Club

The July meeting of Port O' CoCo was a calm celebration of success. We knew by that time that the ***PNW CoCoFEST III*** of a few weeks before was a big success. Just about all the bills have been paid and we are several hundred dollars ahead. Tom Brooks was pleased to see even more money come in from further mug and T-shirt sales. We expect that further sales will pop up now and then in the weeks to come. There is a supply of the "History of Computer Chips" mugs available and few of the very successful FEST III T-shirts. So if you want one, look for the special announcement elsewhere.

Fathoms O' Fun, the community event which we use as the nest for our FEST. We have synchronized our two day event with their two week event for the last three year. This year we formally asked for them to promote the FEST in all of their publicity. They graciously did so. Since they were one of the factors for our success, we wanted to help them with a little seed money for next year.

The meeting was devoted to suggestions on what could be done next year: Whoever does the video taping of the speakers and demos should not be a participant. **Terry Laraway** has done the taping for two years and doing so restricts his getting involved in the event and spending time as he wants. A public address system is a must for the large meetings. We need a map for finding the location and signs for the last couple blocks of the trip. Two years ago we had quite a supply of door prizes. It would be good to have them again for a drawing during the luncheon. We need more personnel for the various tasks of 1) greeter, 2) registration, 3) selling and promoting mugs & T-shirts (or whatever we have), 4) name tags.

For those who may find the registration fee a hardship we could offer for a fee to just cover the food costs and then offer their talents in doing some of the tasks that make a big event like this work. The discount would make it possible for them to attend and at the same time provide us with personnel to allow thing to flow more smoothly.

It was suggested that we have sessions one after the other with no concurrent presentations or have concurrent presentations repeated at least two times so people could see more of what we have available. The former would mean that we would have a limited number of presentations. The latter would allow two or three as many presentations, but require process of prioritizing the presentations by each participant. There would still be some presentations that would have to be missed.

Related to the PNW CoCoFEST III was what to do with the excess revenue. We have talked about doing something outside of the computer community, to reach out of ourselves to make a difference with another part of our society. In the past we briefly talked about a small scholarship for someone in the county. Another idea that has come forth recently: Buy 2-3 modems and install them in members systems for 2 or 3 months for them to tap into the world or telecommunications. Then the modem would be passed onto other members. Slowly we would "wire" the membership. The assumption is that after using a modem for a couple months the member would just have to get one of his/her own. These are just ideas on the table. No conclusions were reached.

Auctioning of tapes, books, printers, paks, and odds and ends contributed by a person in the community who gave his system to the club generated another $40-$50. The rest of the time was divided into small group discussions.

We are talking about having a special "Equipment Fest" for one evening during August. This would be just hands on and nut & bolts for those who are building their system (tower work or chip upgrading, etc.). Rodger Alexander has expressed willingness to make a trip down for this special work party. If you have something you are trying to build and would benefit from a helping hand from several experts, let us know right away! The next regular meeting will be August 16th. An equipment Fest would probably be after that date, but before the beginning of school on a day between Tuesday and Friday.

==Donald Zimmerman==

Seattle 68xxx Micro Users Group

Donald Zimmerman made a quick report on the success of the 3rd Annual PNW CoCoFEST. Don passed out pictures taken of idividuals at the different seminars. A few T-shirts and FEST Mugs are still available.

Rodger Alexander passed out comparative Basic09 and QuickBasic listing of his menuing program that was demonstrated at last month's meeting with the Pioneer Laser Disk Player.

Scott Honaker demonstrated simple surface mount techniques that are simple to use, especially for those who are shy of soldering irons. Scott installed an LED power indicator light in the top of a CoCo-3, but instead of using wires and a standard resistor, he used conductive paint, conductive glue and a surface mount resistor to achieve the entire project. Extremely simple. The added bonus was the elimination of the wire mess required to connect the 5 volts on the mother board in the bottom half of the case up to the LED in the upper half of the case. Scott simply painted the circuit liberally to the edges of the case so when the case was screwed shut the conductive paint made contact between the upper and lower case edges. Case on...LED glows. Case off...LED is dead. Passing his hand mysteriously between the case halfs added to the magic show.

Remainder of the meeting was broken up into small group sessions and exchanging PD Library files.

==Barbara Alexander==

Washington State BBS List

COLUMBIA HTS. BBS
-- Lonview/Kelso --
RiBBS (FidoNET)
(206) 425-5804

DATA WAREHOUSE BBS
-- Spokane --
RiBBS (FidoNET)
(509) 325-6787

BARBEQUED RIBBS
-- Bellingham --
PC-Board (PC-Net) - CoCo Conference #5
(206) 676-5787

OS-9 TACOMA BBS
-- Tacoma --
RiBBS (FidoNET)
(206) 566-8857

ULTIMATE EXPERIENCE BBS
-- Anacortes --
RiBBS (MaxNET)
(206) 299-0491

OS-9 Newsletter
3404 Illinois Lane
Bellingham, WA 98226-4238